In the Claims:

Please amend claims 1, 4-7, 10, 12-14, 16, 18, 22-25, 27, 29, 31-33, 36, 38-39, 41, 43-44, 47, 50-51, 53-54, and 56-65, as indicated below.

1. (Currently amended) A method implemented in a device supporting a cryptography application, ~~for operating a processor~~ the method comprising:

in response to executing a single arithmetic instruction,

multiplying a first number by a second number; and

adding implicitly a partial result from a previously executed single arithmetic instruction to generate a result that represents the first number multiplied by the second number summed with the partial result, wherein the partial result comprises a high order portion of a result of the previously executed single arithmetic instruction;

storing at least a portion of the generated result; and

using the stored at least a portion of the generated result in a subsequent computation in the cryptography application.

2. (Original) The method as recited in claim 1 further comprising performing the adding of the partial result as part of addition operations performed for the multiplying of the first and second number.

3. (Original) The method as recited in claim 1 wherein the partial result is in redundant number representation.

4. (Currently amended) The method as recited in claim 1, wherein said ~~further comprising performing the~~ adding ~~of~~ the partial result ~~by~~ comprises adding the partial result to a multiplication result of the first and second numbers.

5. (Currently amended) The method as recited in claim 1, wherein said storing at least a portion of the generated result comprises ~~further comprising~~ storing a high order portion of the generated result as a next partial result for use with execution of a subsequent single arithmetic instruction.

6. (Currently amended) The method as recited in claim 5, wherein said storing the high order portion of the generated result comprises ~~further comprising~~ storing the high order portion of the generated result into an extended carry register for use with execution of the subsequent single arithmetic instruction.

7. (Currently amended) The method as recited in claim 6, further comprising retrieving an indication of a current value of the extended carry register by executing another single arithmetic instruction that multiplies a third number by a fourth number[[;]] and that implicitly adds current contents of the extended carry register to generate a second result that represents the third number multiplied by the fourth number summed with the current contents of the extended carry register.

8. (Original) The method as recited in claim 7, wherein a low order portion of the second result contains the indication of the current value of the extended carry register.

9. (Original) The method as recited in claim 7, wherein the third and fourth numbers are zero.

10. (Currently amended) The method as recited in claim 6, further comprising loading the extended carry register with a predetermined value by executing another single arithmetic instruction that multiplies a third number by a fourth number[[;]], and

that implicitly adds a current value of the extended carry register, to generate a result that represents the third number multiplied by the fourth number summed with the current value of the extended carry register and to store it in the extended carry register, thereby loading the extended carry register with the predetermined value.

11. (Original) The method as recited in claim 6, further comprising selecting one of a plurality of extended carry registers as the extended carry register.

12. (Currently) The method as recited in claim 6, further comprising accessing the extended carry register ~~by~~ via at least one of a load instruction and a store instruction ~~executed by the processor~~.

13. (Currently amended) The method as recited in claim 1, wherein the single arithmetic instruction is a single multiply-accumulate instruction; wherein the first and second numbers are specified in the single multiply-accumulate instruction as first and second source registers, and a low order portion of the result is stored in a destination location specified in the single multiply-accumulate instruction.

14. (Currently amended) The method as recited in claim 5, wherein the first and second numbers are n-bit numbers, n being a positive integer, and wherein the high order portion of the ~~addition~~ generated result is an n-bit portion ~~n bits~~.

15. (Original) The method as recited in claim 5 further comprising:

in response to executing the subsequent single arithmetic instruction,

multiplying third and fourth numbers specified by the subsequent single arithmetic instruction and adding implicitly the next partial result to generate a second result that represents the third number multiplied by the fourth number summed with the next partial result.

16. (Currently amended) The method as recited in claim 15, ~~as recited~~ further comprising storing the high order portion of the second result to be implicitly added ~~with~~ in response to executing another subsequent single ~~multiply-accumulate~~ arithmetic instruction.

17. (Original) The method as recited in claim 1 wherein the multiplying and adding are implemented to support XOR operations for binary polynomial fields.

18. (Currently amended) A method implemented in a device supporting a cryptography application, ~~for operating a processor~~ the method comprising:

in response to executing a single arithmetic instruction,

multiplying a first number by a second number;

adding implicitly a partial result from a previously executed single arithmetic instruction, wherein the partial result comprises a high order portion of a result of the previously executed single arithmetic instruction; ~~and~~

adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number;

storing at least a portion of the generated result; and

using the stored at least a portion of the generated result in a subsequent computation in the cryptography application.

19. (Original) The method as recited in claim 18 further comprising performing the adding of the partial result as part of addition performed for the multiplying of the first and second number.

20. (Original) The method as recited in claim 18 wherein the partial result is stored in a redundant number representation.

21. (Original) The method as recited in claim 18 further comprising performing the adding of the third number as part of the addition performed for the multiplying of the first and second number.

22. (Currently amended) The method as recited in claim 18, wherein said ~~further comprising performing the~~ adding ~~of~~ the partial result ~~by~~ comprises adding the partial result after generation of a multiplication result of multiplying the first and second numbers.

23. (Currently amended) The method as recited in claim 18, ~~further comprising~~ wherein said storing at least a portion of the generated result comprises storing a high order portion of the generated result as a next partial multiplication result for use with execution of a subsequent single arithmetic instruction.

24. (Currently amended) The method as recited in claim 23 ~~further comprising~~ wherein said storing the high order portion of the generated result comprises storing the high order portion of the generated result into an extended carry register for use with execution of the subsequent arithmetic instruction.

25. (Currently amended) The method as recited in claim 24, further comprising retrieving an indication of a current value of the extended carry register by executing another single arithmetic instruction that multiplies a fourth number by a fifth number,[[;]] ~~and~~ that implicitly adds current contents of the extended carry register, and that adds a sixth number to generate a second result that represents the fourth number

multiplied by the fifth number summed with the current contents of the extended carry register and the sixth number.

26. (Original) The method as recited in claim 25, wherein a low order portion of the second result contains the indication of the current value of the extended carry register.

27. (Currently amended) The method as recited in claim 24, further comprising loading the extended carry register with a predetermined value by executing another single arithmetic instruction that multiplies a fourth number by a fifth number,[[;]] ~~and~~ that implicitly adds a current value of the extended carry register, and that adds a sixth number, to generate a second result that represents the third number multiplied by the fourth number summed with the current value of the extended-carry register and summed with the sixth number and to store it in the extended carry register, thereby loading the extended carry register with the predetermined value.

28. (Original) The method as recited in claim 24, further comprising selecting one of a plurality of extended carry registers as the extended carry register.

29. (Currently amended) The method as recited in claim 24, further comprising accessing the extended carry register via at least one of a load instruction and a store instruction ~~executed by the processor~~.

30. (Original) The method as recited in claim 24 wherein the second number is implicitly identified in the single arithmetic instruction.

31. (Currently amended) The method as recited in claim 30, further comprising accessing a special register storing the second number via at least one of a load instruction and a store instruction ~~executed by the processor~~.

32. (Currently amended) The method as recited in claim 18, further comprising ~~the processor~~ accessing a special register storing the second number via at least one of a load instruction and a store instruction.

33. (Currently amended) The method as recited in claim 18, wherein the first and third numbers are specified in the single arithmetic instruction as first and second source registers and a low order portion of the generated result is stored in a destination location specified in the single arithmetic instruction.

34. (Original) The method as recited in claim 33 wherein the partial result from a previously executed single arithmetic instruction is implicitly specified by the single arithmetic instruction and wherein the second number is explicitly specified by a third source register in the single arithmetic instruction.

35. (Original) The method as recited in claim 33 wherein the partial result from a previously executed single arithmetic instruction is implicitly specified by the single arithmetic instruction and the second number is implicitly specified by the single arithmetic instruction.

36. (Currently amended) The method as recited in claim 23 further comprising:

in response to executing the subsequent single arithmetic instruction,

multiplying a fourth number and a fifth number, the fourth number being specified by the subsequent single arithmetic instruction, ~~and~~ adding implicitly the next partial multiplication result, and adding a sixth number to generate a second result, the second result representing the fourth number multiplied by the fifth number summed with the next partial result and the sixth number.

37. (Original) The method as recited in claim 36 wherein the fifth number and the second number are equal.

38. (Currently amended) The method as recited in claim 36, further comprising storing ~~the~~ a high order portion of the second result to be implicitly added ~~with~~ in response to executing another subsequent single arithmetic instruction.

39. (Currently amended) The method as recited in claim 18, wherein the first number is specified in the single arithmetic instruction in a first source register, ~~and~~ the second number is contained in a special register and is not specified in the single arithmetic instruction, ~~and~~ the third number is specified as a second source register in the single arithmetic instruction, and a low order portion of the generated result is stored in a destination location specified in the single arithmetic instruction.

40. (Original) The method as recited in claim 18 wherein the first, second, and third numbers are specified by source operands in the single arithmetic instruction.

41. (Currently amended) The method as recited in claim 18, wherein a destination location and one of the first number, the second number, and the third number[[s]] ~~and a destination location~~[[,]] are specified by one operand in the single arithmetic instruction.

42. (Original) The method as recited in claim 18 wherein the multiplying and adding operations are implemented for binary polynomial fields.

43. (Currently amended) A processor, comprising an arithmetic circuit, the processor configured to be responsive to execution of a single arithmetic instruction to:

cause the arithmetic circuit to multiply a first number and a second number and to add implicitly a high order portion of a partial result from a previously executed single arithmetic instruction, thereby generating a result that

represents the first number multiplied by the second number summed with the high order portion of the partial result;

store at least a portion of the generated result; and

use the stored at least a portion of the generated result in a subsequent computation.

44. (Currently amended) The processor as recited in claim 43, wherein to store at least a portion of the generated result, the processor is further responsive to the single arithmetic instruction to store a high order portion of the generated result into an extended carry register for use with execution of a subsequent single arithmetic instruction.

45. (Original) The processor as recited in claim 43 wherein the high order portion of the previously executed single arithmetic instruction is stored in a redundant number representation.

46. (Original) The processor as recited in claim 45, wherein the extended carry register is a register accessible via a processor instruction.

47. (Currently amended) The processor as recited in claim 45, wherein the extended carry register has an associated dirty bit indicating whether contents of the extended carry register need to be saved on ~~the~~ a context switch.

48. (Original) The processor as recited in claim 43, wherein the extended carry register is a special register.

49. (Original) The processor as recited in claim 43 wherein the first and second numbers are specified in the single arithmetic instruction as first and second source registers.

50. (Currently amended) A processor, comprising an arithmetic circuit, the processor configured to be responsive to execution of a single arithmetic instruction ~~that upon execution thereof~~ to:

cause[[s]] the arithmetic circuit to multiply a first number and a second number, ~~and~~ to add a third number, and to implicitly add a high order portion of a previous result from a previously executed single arithmetic instruction, thereby generating a result that represents the first number multiplied with the second number, summed with the high order portion of the previous result and with the third number;

store at least a portion of the generated result; and

use the stored at least a portion of the generated result in a subsequent computation.

51. (Currently amended) The processor as recited in claim 50, wherein to store at least a portion of the generated result, the processor is ~~coupled~~ configured to store a high order portion of the generated result for use with execution of a subsequent single arithmetic instruction.

52. (Original) The processor as recited in claim 50 wherein the high order portion of the previous result is stored in a redundant number representation.

53. (Currently amended) The processor ~~as recited in claim~~ as recited in claim 50, wherein the processor is ~~coupled~~ configured to store the high order portion of the generated result into an extended carry register.

54. (Currently amended) The processor as recited in claim 50, wherein the extended carry register is a special register accessible by the processor via at least one of: load instructions and store instructions.

55. (Original) The processor as recited in claim 50, wherein the extended carry register has an associated dirty bit indicating whether contents of the extended carry register need to be saved on a context switch.

56. (Currently amended) The processor as recited in claim 50, wherein the first number is specified in the single arithmetic instruction as a first source register, ~~and~~ the second number is contained in a logically local register and is not specified in the single arithmetic instruction, ~~and~~ the third number is specified as a second source register in the single arithmetic instruction, and a low order portion of the result is stored in a destination location specified in the single arithmetic instruction.

57. (Currently amended) A computer-readable ~~program product encoded on computer readable media, the computer program product comprising~~ storage medium, comprising program instructions executable by a processor to implement a cryptography application:

wherein a single arithmetic instruction in the cryptography application ~~causing~~ causes [[a]] the processor ~~executing the single arithmetic instruction~~ to multiply a first number by a second number and to implicitly add a high order portion of a previously executed single arithmetic instruction to generate a result that represents the first number multiplied with the second number and summed with [[a]] the high order portion of a previously executed single arithmetic instruction,

wherein the single arithmetic instruction further ~~causing~~ causes the processor ~~executing the single arithmetic instruction to keep~~ store a high order

portion of the generated result for use with execution of a subsequent single arithmetic instruction in the cryptography application.

58. (Currently amended) The ~~computer program product~~ storage medium as recited in claim 57, wherein the single arithmetic instruction includes a first source operand and a second source operand, specifying the first number and the second number, and a destination operand[[,]] wherein the single arithmetic instruction further causes ~~causing~~ the processor to store a low order portion of the generated result in a location specified by the destination operand.

59. (Currently amended) The ~~computer program product~~ storage medium as recited in claim 57, ~~further comprising~~ wherein the subsequent single arithmetic instruction ~~causing~~ causes the processor executing the subsequent single arithmetic instruction to multiply a third number by a fourth number and implicitly add the high order portion of the result.

60. (Currently amended) The ~~computer program product~~ storage medium as recited in claim 59, wherein ~~further comprising~~ another single arithmetic instruction in the cryptography application causes ~~causing~~ the processor ~~executing the other single arithmetic instruction~~ to multiply a fifth number by a sixth number and to generate another result without implicitly adding another high order portion of another previously executed result, and to store a high order portion of the other result for use with another subsequent single arithmetic instruction.

61. (Currently amended) A computer-readable ~~program product encoded on computer readable media, the computer program product comprising~~ storage medium, comprising program instructions executable by a processor to implement a cryptography application:

wherein a single arithmetic instruction ~~causing~~ in the cryptography application causes the [[a]] processor ~~executing the single arithmetic instruction~~ to:

multiply a first number by a second number;

add implicitly a partial multiplication result from a previously executed single arithmetic instruction and a third number to generate a result that represents the first number multiplied by the second number summed with the partial multiplication result and summed with the third number; and

store a high order portion of the generated result for use with execution of a subsequent single arithmetic instruction in the cryptography application.

62. (Currently amended) The ~~computer program product~~ storage medium as recited in claim 61, wherein ~~further comprising~~ the subsequent second single arithmetic instruction ~~causing~~ causes the processor ~~executing the subsequent single arithmetic instruction~~ to multiply a fourth number by the second number, ~~and~~ to add a fifth number, and to implicitly add the high order portion of the generated result.

63. (Currently amended) The ~~computer program product~~ storage medium as recited in claim 61, ~~further comprising~~ wherein the subsequent single arithmetic instruction ~~causing~~ causes the processor ~~executing the subsequent single arithmetic instruction~~ to multiply a fourth number by a fifth number, ~~and~~ to add a sixth number, and to implicitly add the high order portion of the generated result.

64. (Currently amended) A processor supporting a cryptography application, comprising:

means, responsive to a single multiply-accumulate instruction in the cryptography application, for multiplying a first number with a second number and implicitly adding a partial result of a previously executed single multiply-

accumulate instruction to generate a .result that represents the first number multiplied by the second number summed with the partial result; and

means for storing a high order portion of the result for use with execution of a subsequent single multiply-accumulate instruction in the cryptography application.

65. (Currently amended) A processor supporting a cryptography application, comprising:

means, responsive to a single multiply-accumulate instruction in a cryptography application, for multiplying a first number with a second number, ~~and~~ for implicitly adding a partial result of a previously executed single multiply-accumulate instruction, and for adding a third number to generate a result that represents the first number multiplied by the second number summed with the partial result and the third number; and

means for storing a high order portion of the generated result for use with execution of a subsequent multiply-accumulate instruction in the cryptography application.